Informe GreenMart_Dataset

Informe sobre limpieza y normalización de datos- GreenMart Retail

1) **Carga y exploración del dataset**

```
df = pd.read_csv("greenmart_customers_products.csv")
df.head()
df.info()
df.describe()
df.columns
df.isnull().sum()
```

El objetivo es conocer la estructura que tenemos en nuestro dataset. Identificamos el número de filas y columnas, los tipos de datos y además, los valores faltantes. Ésto nos permite revisar estadísticas básicas de las columnas numéricas.

Es importante saber qué tipo de datos tenemos y dónde hay problemas o posibles problemas.

2) **Eliminación de columnas con muchos valores faltantes**

```
cols_to_drop = ["customer_name", "age", "city"]
df = df.drop(columns=[col for col in cols_to_drop if col in df.columns], errors="ignore")
```

Eliminamos las columnas que tienen demasiados valores nulos. En el caso de customer_name y city, no aportaban información estadística útil. Las columnas age y purchase_date tenían muchos valores faltantes, eliminarlas nos permite reducir el ruido y el tamaño del dataset. Cuando una columna tiene más del 50-60% de valores faltantes, se considera no fiable. Por ende, la mejor opción es la eliminación de las mismas.

3) **Eliminación de duplicados**

```
df= df.drop_duplicates()
```

De esta forma, garantizamos que cada registro sea único. Evitamos distorsionar los análisis estadísticos.

4) **Normalización de fechas**

```
df["purchase_date"] = pd.to_datetime(df["purchase_date"], errors='coerce')
df["year"] = df["purchase_date"].dt.year
df["month"] = df["purchase_date"].dt.month
df["day"] = df["purchase_date"].dt.day
df["weekday"] = df["purchase_date"].dt.weekday
df = df.drop(columns=["purchase_date"])
```

Convertimos la fecha de string a un dato manipulable. Extraemos los componentes útiles: año, mes, día y día de la semana. Procedemos a la eliminación de la columna original, ésto permite un análisis posterior por mes, día, etc. Además, reduce el tamaño del dataset.

5) **Normalización de valores numéricos**

```
num_cols = [col for col in ["purchase_quantity", "price_per_unit", "total_spent"] if col in
df.columns]
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
```

Escalamos las columnas numéricas para que tengan media =0 y desviación=1. Es importante que todas las variables numéricas estén en la misma escala, evitamos que los valores más grandes dominen los resultados.

6) **Codificación de variables categóricas**

```
cols_to_encode = [col for col in ["product_name", "category"] if col in df.columns]
le = LabelEncoder()
for col in cols_to_encode:
    df[col] = le.fit_transform(df[col])
```

Convertir las categorías en números únicos para su interpretación.
Se eligió Label Encoding en lugar de On-Hot Encoding ya que algunas columnas(product_name) tienen muchos valores y ésto, aumentaba el tamaño del dataset.

7) **Guardado del dataset limpio**

```
df.to_csv("Greenmart_dataset_limpio.csv", index=False)
files.download("Greenmart_dataset_limpio.csv")
```

Permite tener un archivo final listo para análisis.

Dificultades encontradas en los datos:

- Columnas con muchos valores faltantes: se decidió eliminarlas
- Columnas casi únicas(customer_name): no se pueden codificar con One Hot Encoding, inflan el tamaño del dataset
- Fechas como strings: convertirlas a datetime y extraer información
- Posibles errores al escalar o codificar: algunas columnas fueron eliminadas antes de procesarlas, así que hubo que filtrar solo las existentes para evitar errores.